Indovina la Città

Indice:

Introduzione:

“Indovina la Città” è un eccitante gioco multiplayer che mette alla prova le conoscenze dei giocatori sulle città di tutto il mondo. Con un'interfaccia semplice e coinvolgente, il gioco offre un'esperienza divertente e competitiva mentre i giocatori si sfidano a indovinare il nome delle città basandosi su immagini visualizzate.

Abstract (Eng):

“Indovina la Città” is a multiplayer web application that challenges players to guess the name of the city based on displayed images. This document provides a detailed overview of the features, rules, and key elements of the game, as well as outlining its winning condition and end game. The game utilizes WebSocket to ensure real-time communication between players and the server, allowing for a smooth and interactive experience. Users connect to the server to start the game and are presented with images of cities, each followed by four answer options. Players must select the name of the city corresponding to the image within a time limit, with the score increasing for each correct answer. Key elements of the game include dynamic display of city images and names, a countdown timer for each puzzle, and score management to determine the winner at the end of the game. The player with the highest score is declared the winner, encouraging accuracy and quick response. “Indovina la Città” offers an engaging and competitive experience, promoting social interaction and challenge among players. Once the game is over, the final score is displayed, and the winner is announced, providing a satisfying conclusion and encouraging players to return for new challenges.

Abstract (Ita):

Il gioco "Capolavoro: Indovina la Città" è un'applicazione web multiplayer che sfida i giocatori a indovinare il nome della città basandosi su immagini visualizzate. Questo documento fornisce una panoramica dettagliata delle caratteristiche, delle regole e degli elementi chiave del gioco, oltre a delineare la sua modalità di vittoria e la fine del gioco.Il gioco si avvale di WebSocket per garantire una comunicazione in tempo reale tra i giocatori e il server, consentendo un'esperienza fluida e interattiva.

Gli utenti si connettono al server per iniziare il gioco e vengono presentate immagini di città, ciascuna seguita da quattro opzioni di risposta. I giocatori devono selezionare il nome della città corrispondente all'immagine entro un tempo limite, con il punteggio che aumenta per ogni risposta corretta.Gli elementi chiave del gioco includono la visualizzazione dinamica delle immagini e dei nomi delle città, un contatore del tempo rimanente per ogni indovinello e la gestione del punteggio per determinare il vincitore alla fine del gioco. Il giocatore con il punteggio più alto è dichiarato vincitore, incentivando la precisione e la rapidità di risposta."Capolavoro: Indovina la Città" offre un'esperienza coinvolgente e competitiva, che promuove l'interazione sociale e la sfida tra i giocatori. Conclusa la partita, viene mostrato il punteggio finale e il vincitore viene annunciato, fornendo una conclusione soddisfacente e incoraggiando i giocatori a tornare per nuove sfide.In sintesi, il gioco "Capolavoro: Indovina la Città" rappresenta un'opportunità per i giocatori di mettere alla prova le proprie conoscenze geografiche in un ambiente divertente e stimolante, favorito da una comunicazione in tempo reale e da un sistema di punteggio che premia la precisione e la velocità di risposta.

Obiettivo del Gioco:

Il gioco "Indovina la Città" è un'applicazione web multiplayer che sfida i giocatori a indovinare il nome della città basandosi su immagini visualizzate. I giocatori competono per ottenere il punteggio più alto indovinando correttamente il nome della città mostrata.

Regole del Gioco:

1. All'inizio del gioco, i giocatori si connettono tramite WebSocket al server.
2. Viene visualizzata un'immagine di una città nel centro della schermata.
3. Quattro pulsanti con nomi di città diverse vengono mostrati sotto l'immagine.
4. I giocatori devono fare clic sul pulsante che corrisponde al nome della città visualizzata.
5. Ogni risposta corretta incrementa il punteggio del giocatore.
6. Il gioco procede mostrando immagini di città diverse a intervalli regolari.
7. Il tempo per indovinare è limitato, e il contatore viene visualizzato sulla schermata.
8. Dopo un certo numero di immagini visualizzate o quando il tempo si esaurisce, viene mostrato il punteggio finale.
9. Il giocatore con il punteggio più alto alla fine vince.

Elementi Chiave del Codice:

- Utilizzo di WebSocket per la comunicazione in tempo reale tra i giocatori e il server.
- Visualizzazione dinamica di immagini e nomi di città.
- Contatore che tiene traccia del tempo rimanente per ogni indovinello.
- Gestione del punteggio e determinazione del vincitore alla fine del gioco.

Modalità di Vittoria:

Il giocatore che ottiene il punteggio più alto alla fine del gioco è dichiarato vincitore.

Fine del Gioco:

Il gioco termina dopo un certo numero di indovinelli o quando il tempo limite scade. Viene visualizzato il punteggio finale e il vincitore viene annunciato.

Descrizione del Codice:

Il codice è strutturato in modo modulare e composto da due parti principali:
il frontend e il backend.

Frontend:

Il frontend è implementato utilizzando HTML, CSS e JavaScript.

- HTML: Il file HTML definisce la struttura della pagina web del gioco, inclusi gli elementi per visualizzare l'immagine della città, i pulsanti per le risposte, il contatore del tempo e il punteggio.
- CSS: Lo stile CSS viene utilizzato per definire l'aspetto visivo della pagina, inclusi colori, dimensioni e posizionamento degli elementi.
- JavaScript: Il codice JavaScript gestisce la logica del gioco, inclusi gli aggiornamenti dell'immagine della città, la gestione dei clic sui pulsanti, il contatore del tempo e la comunicazione con il server WebSocket. Viene utilizzata anche la libreria WebSocket per stabilire e gestire la connessione con il server.

Backend:

Il backend è implementato utilizzando Node.js e il framework Express per creare un server HTTP, e il modulo ws per gestire il server WebSocket.

- Server HTTP: Il server HTTP gestisce le richieste dei client per caricare la pagina HTML del gioco.
- Server WebSocket: Il server WebSocket gestisce la comunicazione in tempo reale tra i client e il server. Gestisce la connessione e la disconnessione dei giocatori, invia informazioni sulla connessione e sul punteggio ai client, e confronta i punteggi dei giocatori alla fine del gioco per determinare il vincitore.

Comunicazione WebSocket:

La comunicazione tra il frontend e il backend avviene tramite WebSocket. Quando un giocatore si connette, il server WebSocket invia informazioni sulla connessione agli altri giocatori. Durante il gioco, il server invia l'immagine della città, aggiorna il contatore del tempo e riceve le risposte dei giocatori. Alla fine del gioco, confronta i punteggi dei giocatori e invia il risultato a tutti i client.

Gestione del Tempo e del Punteggio:

Il contatore del tempo viene aggiornato ogni secondo e quando raggiunge zero, cambia l'immagine della città. I punteggi dei giocatori vengono incrementati quando indovinano correttamente il nome della città. Alla fine del gioco, il server confronta i punteggi e determina il vincitore.

Vantaggi:

Questo approccio consente un'esperienza di gioco fluida e coinvolgente per i giocatori, con aggiornamenti in tempo reale dell'immagine della città, del contatore del tempo e del punteggio. La comunicazione bidirezionale tramite WebSocket consente una sincronizzazione efficiente tra il frontend e il backend, consentendo una gestione rapida e affidabile del gioco multiplayer.

Il codice backend:

```
// Richiamo il modulo fs per la gestione dei file, express per il framework web e path
per la gestione dei percorsi
var fs = require('fs');
const express = require('express');
const path = require('path');

// Creo un'applicazione Express e un server HTTP basato su di essa
const server = express();
const http = require('http').Server(server);

// Configuro una rotta per la radice del server, che invia il file HTML
server.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'capolavoro.html'));
});

// Avvio del server HTTP sulla porta 3000
http.listen(3000, () => {
  console.log('Listening on 3000');
});

// Importo il modulo WebSocket (ws)
const { Server } = require('ws');

// Variabili per tenere traccia del numero di giocatori, client WebSocket e caselle
let quanti = 0;
let clients = [];
let punteggiRicevuti = {};
let risultatoConfronto = null; // Inizializza la variabile risultatoConfronto
let punteggiFinali = {};


for (let i = 1; i <= 8; i++) {
  punteggiFinali[i] = null; // Inizializza tutti i punteggi finali a null
  punteggiRicevuti[i] = false; // Inizializza tutti i punteggi ricevuti a false
}

// Creo un server WebSocket separato dal server HTTP
const ws_server = new Server({ noServer: true });

// Gestione dell'aggiornamento del protocollo HTTP a WebSocket
http.on('upgrade', (request, socket, head) => {
  ws_server.handleUpgrade(request, socket, head, (socket) => {
    ws_server.emit('connection', socket, request);
  });
});
```

```
// Evento di connessione WebSocket
ws_server.on('connection', (ws) => {
  quanti++;

  // Se ci sono già più di un giocatore, termina la connessione del nuovo arrivato
  if (clients.length > 7) {
    quanti--;
    ws.terminate();
  }

  // Assegna un ID univoco al WebSocket
  ws.id = quanti;
  clients.push(ws.id);

  // Stampa il nuovo giocatore e gli ID dei giocatori esistenti
  console.log("nuovo:" + ws.id);
  let s = "";
  for (let z = 0; z < clients.length; z++) s = s + clients[z] + " ";
  console.log("clients : " + s);

  // Invia informazioni sulla connessione a tutti i client WebSocket
  let position = {
    quanti: clients.length,
    tipo: 0
  }
  let data = JSON.stringify({ 'position': position });
  ws_server.clients.forEach((client) => {
    client.send(data);
  });

  // Invia un messaggio al client appena connesso con le informazioni sulla sua
connessione
  position = {
    quanti: clients.length,
    chi: ws.id,
    tipo: -1
  }
  data = JSON.stringify({ 'position': position });
  ws.send(data);

  // Inizializza il punteggio ricevuto per questo client come falso
  punteggiRicevuti[ws.id] = false;

  // Gestisce l'evento di chiusura della connessione WebSocket
  ws.on('close', () => {
    // Rimuove il giocatore che si è disconnesso dalla lista dei giocatori
    for (let k = 0; k < clients.length; k++) {
```

```
    if (clients[k] === ws.id) {
      clients.splice(k, 1);
    }
  }

  // Rimuove il punteggio ricevuto per questo client
  delete punteggiRicevuti[ws.id];

  // Stampa l'ID del giocatore che si è disconnesso e gli ID dei giocatori rimanenti
  console.log("esce:" + ws.id);
  s = "";
  for (let z = 0; z < clients.length; z++) s = s + clients[z] + " ";
  console.log("clients : " + s);

  // Invia informazioni sulla disconnessione a tutti i client WebSocket rimanenti
  position = {
    quanti: clients.length,
    tipo: -2
  }
  const data = JSON.stringify({ 'position': position });
  ws_server.clients.forEach((client) => {
    client.send(data);
  });
 });

 ws.on('message', (message) => {
  // Codice per gestire i messaggi WebSocket
  const parsedData = JSON.parse(message);
  if (parsedData && parsedData.punteggioFinale !== undefined &&
!punteggiRicevuti[ws.id]) {
    // Esegui le operazioni necessarie con il punteggio finale
    console.log('Punteggio finale ricevuto dal client ' + ws.id + ':',
parsedData.punteggioFinale);

    // Salva il punteggio finale per questo client
    punteggiFinali[ws.id] = parsedData.punteggioFinale;

    // Dopo aver salvato il punteggio finale per un client
    console.log('Punteggio finale salvato per il client ' + ws.id + ':',
punteggiFinali[ws.id]);

    // Imposta il punteggio ricevuto per questo client su true per indicare che è stato
ricevuto
    punteggiRicevuti[ws.id] = true;

    // Controlla se tutti i giocatori hanno inviato i loro punteggi finali
    let tuttiPunteggiRicevuti = true;
    for (let i = 1; i <= clients.length; i++) {
```

```
        if (!punteggiRicevuti[i]) {
          tuttiPunteggiRicevuti = false;
          break;
        }
      }

      if (tuttiPunteggiRicevuti) {
        // Confronta i punteggi e determina il vincitore
        let vincitore = null;
        let punteggioMassimo = -Infinity;
        for (let i = 1; i <= clients.length; i++) {
          if (punteggiFinali[i] > punteggioMassimo) {
            punteggioMassimo = punteggiFinali[i];
            vincitore = i;
          }
        }

        // Controlla se ci sono più vincitori con lo stesso punteggio massimo
        let vincitoriMultipli = [];
        for (let i = 1; i <= clients.length; i++) {
          if (punteggiFinali[i] === punteggioMassimo) {
            vincitoriMultipli.push(i);
          }
        }
        console.log("vince " + vincitore);
        // Invia il risultato del confronto a tutti i client
        const risultatoConfronto = JSON.stringify({ 'vincitore': vincitoriMultipli });
        ws_server.clients.forEach((client) => {
          client.send(risultatoConfronto);
        });
        // Dopo aver inviato il risultato del confronto a tutti i client
        console.log('Risultato del confronto inviato a tutti i client:', risultatoConfronto);


        // Resettare i punteggi ricevuti per prepararsi a una nuova partita
        for (let i = 1; i <= clients.length; i++) {
          punteggiRicevuti[i] = false;
          punteggiFinali[i] = null;
        }
      }
    }
  });
});
```

Questo codice gestisce la connessione WebSocket, la comunicazione tra i client e il server, e il confronto dei punteggi alla fine del gioco.

Il codice frontend:

```
<html>

<head>
  <!-- Definizione dello stile CSS -->
  <style type="text/css">
    .background {
      background-image:
url('https://t4.ftcdn.net/jpg/02/24/03/21/360_F_224032170_7PfrDJYWjCw4Rs1WFvhPkiSPuD02sw1q.jpg');
      background-size: cover;
      /* Per coprire l'intero elemento con l'immagine */
      color: white;
      /* Colore del testo */
      padding: 20px;
      text-align: center;
      /* Per centrare il testo */
    }

    .oggetto {
      background-color: blue;
      color: white;
      height: 50px;
      width: 50px;
      text-align: center;
      font-size: 20px;
      border-width: 1px;
      border: solid black;
      float: left;
      margin: 5px;
      visibility: hidden;
    }

    .connessioni:hover {
      background-color: #080808;
    }
    .punteggio:hover {
      background-color: #080808;
    }

    .client:hover {
      background-color: #080808;
    }

    .connessioni {
```

```
  position: absolute;
  top: 25px;
  left: 10px;
  background-color: rgb(64, 64, 65);
  color: rgb(253, 252, 252);
  height: 50px;
  width: 150px;
  text-align: center;
  font-size: 20px;
  border-width: 1px;
  border: solid black;
  margin: 5px;
  visibility: hidden;
}

.client {
  position: absolute;
  top: 25px;
  left: 170px;
  background-color: rgb(64, 64, 65);
  color: rgb(252, 252, 252);
  height: 50px;
  width: 150px;
  text-align: center;
  font-size: 20px;
  border-width: 1px;
  border: solid black;
  margin: 5px;
  visibility: hidden;
}

.nonconn {
  position: absolute;
  top: 25px;
  left: 10px;
  background-color: pink;
  color: black;
  height: 50px;
  width: 370px;
  text-align: center;
  font-size: 20px;
  border-width: 1px;
  border: solid black;
  margin: 5px;
  visibility: visible;
  transform: translate(-50%, -50%);
  top: 50%;
  left: 50%;
```

```
}

.centro {
  position: absolute;
  top: 40%;
  left: 50%;
  transform: translate(-50%, -50%);
  background-color: rgb(253, 252, 250);
  color: white;
  height: 300px;
  width: 450px;
  text-align: center;
  font-size: 20px;
  border-width: 1px;
  border: solid black;
}

.centro img {
  width: 100%;
  height: 100%;
  object-fit: cover;
  /* Per mantenere le proporzioni dell'immagine */
}

.counter {
  position: absolute;
  transform: translate(-50%, -50%);
  top: 20%;
  left: 50%;
  font-size: 20px;

}

.h1 {
  position: absolute;
  transform: translate(-50%, -50%);
  top: 5%;
  left: 50%;
  font-size: 50px;
  display: inline-block;
  font-family: verdana;

}

.azione-button {
  position: absolute;
  height: 75px;
  width: 125px;
```

```
    transform: translate(-50%, -50%);
    font-size: 50px;
    margin: 10px;
    padding: 10px;
    font-size: 16px;
    color: white;
    cursor: pointer;
    border-radius: 5px;
    font-size: 22px;
  }

  .azione-button:hover {
    background-color: #2b632d;
  }

  .punteggio {
    text-align: center;
    position: absolute;
    transform: translate(-50%, -50%);
    top: 45%;
    left: 50%;
    font-size: 30px;
    height: 250px;
    width: 350px;
    border: solid black;
    background-color: rgb(20, 20, 20);
    visibility: hidden;
    animation: rainbow 10s infinite;
  }

  @keyframes rainbow {
  0% {color: red;}
  10% {color: orange;}
  20% {color: yellow;}
  30% {color: green;}
  40% {color: blue;}
  50% {color: indigo;}
  60% {color: violet;}
  70% {color: pink;}
  80% {color: purple;}
  90% {color: cyan;}
  100% {color: red;}
}
  @keyframes rainbow2 {
  0%  {background-color: blue;}
  10% {background-color: indigo;}
  20% {background-color: blue;}
  30% {background-color: purple;}
```

```
    40% {background-color: indigo;}
    50% {background-color: blue;}
    60% {background-color: purple;}
    70% {background-color: indigo;}
    80% {background-color: blue;}
    90% {background-color: indigo;}
    100% {background-color: purple;}

  }
  @keyframes rainbow3 {
    0%  {background-color: purple;}
    10% {background-color: blue;}
    20% {background-color: indigo;}
    30% {background-color: purple;}
    40% {background-color: blue;}
    50% {background-color: indigo;}
    60% {background-color: purple;}
    70% {background-color: blue;}
    80% {background-color: indigo;}
    90% {background-color: purple;}
    100% {background-color: blue;}



  }

    .h1 span {
    animation: rainbow 10s infinite;
  }

  .rainbow-button2 {
    animation: rainbow2 20s infinite alternate;
    transition: box-shadow 0.3s;
  }
  .rainbow-button3 {
    animation: rainbow3 20s infinite alternate;
    transition: box-shadow 0.3s;
  }
  .rainbow-button2:hover {
    box-shadow: 0 4px 8px rgb(255, 255, 255);
  }
  .rainbow-button3:hover {
    box-shadow: 0 4px 8px rgba(255, 255, 255);
  }
  </style>
</head>

<body class="background">
```

```
  <!-- Intestazione del gioco -->
  <h1 class="h1">
    <span style="color: red;">I</span>
    <span style="color: orange;">N</span>
    <span style="color: yellow;">D</span>
    <span style="color: green;">O</span>
    <span style="color: blue;">V</span>
    <span style="color: indigo;">I</span>
    <span style="color: violet;">N</span>
    <span style="color: pink;">A</span>
    <span style="color: purple;"> L</span>
    <span style="color: cyan;">A</span>
    <span style="color: red;"> C</span>
    <span style="color: orange;">I</span>
    <span style="color: yellow;">T</span>
    <span style="color: green;">T</span>
    <span style="color: blue;">À</span>
   </h1>

  <!-- Elementi HTML per visualizzare le connessioni e i giocatori -->
  <div id='B' class='connessioni'></div>
  <div id='C' class='client'></div>
  <div id='D' class='nonconn'>Troppe connessioni</div>

  <!-- Elemento HTML per visualizzare l'immagine al centro -->
  <div class='centro' id='centroDiv'></div>

  <!-- Elemento HTML per visualizzare il contatore del tempo -->
  <div id='counter' class='counter'></div>

  <!-- Elemento HTML per visualizzare il punteggio finale -->
  <div id='punteggio' class='punteggio'></div>

  <div id='p1' class='punteggio rainbow'></div>

  <!-- Pulsanti per le azioni (indovinare la città) -->
  <button class="azione-button rainbow-button3" id="n1" style="top: 65%; left: 45%;"
onclick="eseguiAzione('dubai')">Dubai</button>
  <button class="azione-button rainbow-button2" id="n2" style="top: 65%; left: 54%;"
onclick="eseguiAzione('roma')">Roma</button>
  <button class="azione-button rainbow-button2" id="n3" style="top: 75%; left: 45%;"
onclick="eseguiAzione('barcellona')">Barcellona</button>
  <button class="azione-button rainbow-button3" id="n4" style="top: 75%; left: 54%;"
onclick="eseguiAzione('parigi')">Parigi</button>



  <!-- Script JavaScript per la logica del gioco -->
```

```
  <script>
    // Inizializzazione della connessione WebSocket
    let webSocket = new WebSocket(`ws://${location.host}`);
    // Variabili per il punteggio, la città corrente, le immagini visualizzate e il tempo
rimanente
    let punteggioAttuale = 0;
    let cittàCorrente;
    let immaginiVisualizzate = [];
    let tempoRimanente = 0; // inizializza il contatore a 0 secondi
    let punteggioFinale = 0;

    // Gestore degli eventi quando viene ricevuto un messaggio dalla WebSocket
    webSocket.onmessage = (event) => {
      // Parsa il messaggio JSON ricevuto
      const data = JSON.parse(event.data); //è utilizzata per convertire una stringa
JSON ricevuta attraverso un evento WebSocket in un oggetto JavaScript.

      // Ottieni i riferimenti agli elementi HTML utilizzati nel gioco
      const eb = document.getElementById('B');
      const ec = document.getElementById('C');
      const ed = document.getElementById('D');
      const centroDiv = document.getElementById('centroDiv');
      const counterDiv = document.getElementById('counter');
      const p = document.getElementById('punteggio');
      const b1 = document.getElementById('n1');
      const b2 = document.getElementById('n2');
      const b3 = document.getElementById('n3');
      const b4 = document.getElementById('n4');

      // Verifica se il messaggio contiene il vincitore
      if (data.vincitore !== undefined) {
        console.log(data.vincitore);
        // Visualizza il punteggio finale e i vincitori
        const p1 = document.getElementById('p1');
        if (data.vincitore.length > 1) {
          const vincitori = data.vincitore.map((vincitore) => `GIOCATORE
${vincitore}`).join(', ');
          p1.innerHTML = `<br>PUNTEGGIO FINALE:
${punteggioAttuale}<br><br>VINCITORI:<br>${vincitori}<br><br>LA PARTITA È
FINITA`;
        } else if (data.vincitore > 0 && data.vincitore.length <= 1) {
          const vincitori = data.vincitore.map((vincitore) => `GIOCATORE
${vincitore}`).join(', ');
          p1.innerHTML = `<br>PUNTEGGIO FINALE:
${punteggioAttuale}<br><br>VINCITORE:<br>${vincitori}<br><br>LA PARTITA È
FINITA`;
        } else {
```

```
            p1.innerHTML = `<br>PUNTEGGIO FINALE:
${punteggioAttuale}<br><br>NESSUN VINCITORE<br><br>LA PARTITA È FINITA`;
          }
          p1.style.visibility = 'visible';
          p.style.visibility = 'hidden';
        }




        // Gestisci diversi tipi di messaggi ricevuti
        if (data.position.tipo && data.position.tipo === 999) {

          // Nascondi gli altri elementi
          counterDiv.style.visibility = 'hidden';
          centroDiv.style.visibility = 'hidden';
          b1.style.visibility = 'hidden';
          b2.style.visibility = 'hidden';
          b3.style.visibility = 'hidden';
          b4.style.visibility = 'hidden';
          return;
        }

        if (data.position.tipo == 0) {
          // Messaggio di connessione
          const cosa = data.position.cosa;
          const quanti = data.position.quanti;

          eb.innerHTML = "Connessi:" + quanti;
          eb.style.visibility = "visible";
          ed.style.visibility = "hidden";
        }

        if (data.position.tipo == -1) {
          // Messaggio di giocatore
          const chi = data.position.chi;
          const quanti = data.position.quanti;
          ec.innerHTML = "Giocatore:" + chi;
          ec.style.visibility = "visible";
        }

        if (data.position.tipo == -2) {
          // Messaggio di aggiornamento connessioni
          const quanti = data.position.quanti;
          eb.innerHTML = "Connessi:" + quanti;
        }

        // Cambia l'immagine al centro ogni 10 secondi
```

```
        const immagini = [
          { url:
"https://arenatours.com/wp-content/uploads/united-arab-emirates/tour-por-la-ciudad-de-dubai/tour-por-la-ciudad-de-dubai-0-0.jpeg", città: 'dubai' },
          { url:
"https://siviaggia.it/wp-content/uploads/sites/2/2021/03/laghi-love-dubai.jpeg", città:
'dubai' },
          { url:
"https://media-assets.vanityfair.it/photos/651286895935f9b24bf062d1/16:9/w_1280,c_limit/Large-Dubai%20Creek%201R9A8165_Floor.jpg", città: 'dubai' },
          { url:
"https://cdn-imgix.headout.com/collection-card-image/158/image/09c609a1134ea7b034ee4cbd61bdbba5-158-dubai-002-dubai-burj-khalifa-01.jpg", città: 'dubai' },
          { url:
"https://biglietti.roma.it/wp-content/uploads/sites/131/fontana-trevi-vasca-hd.jpg",
città: 'roma' },
          { url: "https://www.noidiroma.com/wp-content/uploads/2017/10/il-vaticano.jpg",
città: 'roma' },
          { url:
"https://upload.wikimedia.org/wikipedia/commons/thumb/5/5f/Stadio_Olimpico_2024.jpg/1200px-Stadio_Olimpico_2024.jpg", città: 'roma' },
          { url:
"https://www.expedia.it/stories/wp-content/uploads/2021/06/15970-share-image.jpg",
città: 'roma' },
          { url: "https://fermoiltempoeviaggio.it/wp-content/uploads/2022/07/Bacio.jpg",
città: 'barcellona' },
          { url: "https://attrazionibarcellona.it/images/tickets_camp_nou.jpg", città:
'barcellona' },
          { url:
"https://www.limprenditore.com/wp-content/uploads/2019/02/3.Modello-Barcellona-1932x966.jpg", città: 'barcellona' },
          { url:
"https://www.meteoweb.eu/wp-content/uploads/2014/07/SAGRADA-FAMILIA-COP.jpg",
città: 'barcellona' },
          { url:
"https://upload.wikimedia.org/wikipedia/commons/6/66/Louvre_Museum_Wikimedia_Commons.jpg", città: 'parigi' },
          { url:
"https://www.raccontaviaggi.it/wp-content/uploads/2022/09/Cose_principali_da_vedere_a_Parigi_Arco_di_Trionfo.jpg", città: 'parigi' },
          { url:
"https://img-prod.tgcom24.mediaset.it/images/2022/04/05/133041574-7a84a225-ef6d-482c-a459-fb9f5e79d89e.jpg", città: 'parigi' },
          { url:
"https://hips.hearstapps.com/hmg-prod/images/the-eiffel-tower-is-seen-before-the-lights-are-switched-off-news-photo-1652029958.jpg", città: 'parigi' }
        ];
```

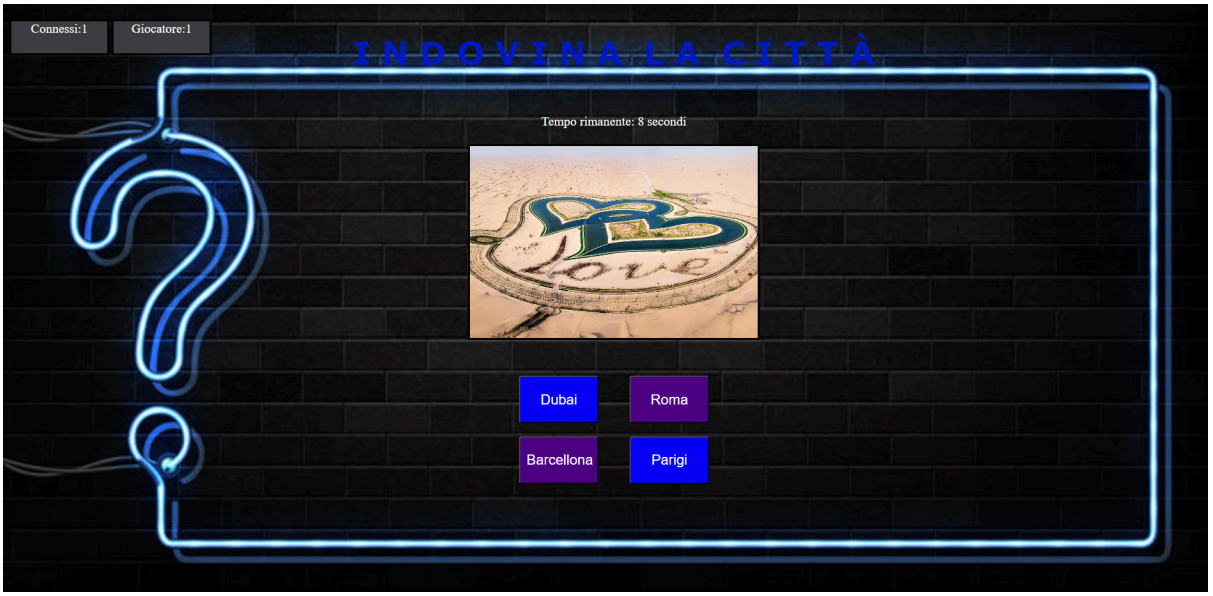
```
        // Funzione per cambiare l'immagine visualizzata
        function cambiaImmagine() {
          // Se tutte le immagini sono state visualizzate almeno una volta, ferma la
visualizzazione
          if (immaginiVisualizzate.length === immagini.length) {
            clearInterval(counterInterval);
            counterDiv.style.visibility = "hidden";
            centroDiv.style.visibility = "hidden";
            b1.style.visibility = "hidden";
            b2.style.visibility = "hidden";
            b3.style.visibility = "hidden";
            b4.style.visibility = "hidden";
            mostraPunteggioFinale();
            return;
          }

          // Seleziona una nuova immagine in modo casuale
          let numeroCasuale;
          do {
            numeroCasuale = Math.floor(Math.random() * immagini.length);
          } while (immaginiVisualizzate.includes(numeroCasuale));

          // Aggiungi l'immagine visualizzata alla lista
          immaginiVisualizzate.push(numeroCasuale);
          cittàCorrente = immagini[numeroCasuale].città;
          centroDiv.innerHTML = `<img src="${immagini[numeroCasuale].url}"
alt="Immagine di ${cittàCorrente}">`;
        }

        // Funzione per mostrare il punteggio finale
        function mostraPunteggioFinale() {
          punteggioFinale = punteggioAttuale;
          p.innerHTML = `<br>PUNTEGGIO FINALE: ${punteggioFinale}<br><br>IN
ATTESA DEGLI ALTRI GIOCATORI...`;
          p.style.visibility = 'visible';
          inviaPunteggioFinale();
        }

        function inviaPunteggioFinale() {
          const data = JSON.stringify({ punteggioFinale: punteggioFinale });
          webSocket.send(data);
        }

        // Funzione per aggiornare il contatore del tempo
        function aggiornaContatore() {
          counterDiv.innerHTML = `Tempo rimanente: ${tempoRimanente} secondi`;
          tempoRimanente = tempoRimanente - 0.5;
```

```
      // Quando il tempo arriva a 0, cambia l'immagine
      if (tempoRimanente < 0) {
        tempoRimanente = 20; // resetta il contatore a 10 secondi quando arriva a 0
        cambiaImmagine();
      }
    }

    // Avvia la funzione di aggiornamento del contatore ogni secondo
    const counterInterval = setInterval(aggiornaContatore, 1000);
  };


  // Funzione chiamata quando viene cliccato uno dei pulsanti di azione
  function eseguiAzione(cittaCliccata) {

    // Se la città cliccata è corretta, incrementa il punteggio e azzera il tempo
    if (cittaCliccata === cittàCorrente) {
      punteggioAttuale++;
      tempoRimanente = 0;
    } else {
      tempoRimanente = 0;
    }
  }


  </script>
</body>

</html>
```

Questo codice HTML definisce la struttura della pagina del gioco "Indovina la Città" e include gli elementi per visualizzare connessioni, giocatori, immagini, contatore del tempo, punteggio e pulsanti per le azioni. Il codice JavaScript presente all'interno dello script gestisce la logica del gioco, incluso il cambio di immagini, il conteggio del tempo e la gestione delle risposte dei giocatori.

Videate del gioco:

Schermata Client 1(Giocatore 1):



Schermata Client 2(Giocatore 2):

Schermata Finale:

La documentazione completa del gioco "Indovina la Città" offre una visione dettagliata di tutte le sue componenti, inclusi sia il frontend che il backend. Le videate aggiuntive forniscono un'illustrazione visiva del gioco in azione, facilitando la comprensione del suo funzionamento. Dal collegamento dei giocatori al server WebSocket alla gestione del punteggio e alla determinazione del vincitore, ogni aspetto del gioco è stato chiarito in modo esaustivo. Grazie a questa documentazione, è possibile ottenere una visione chiara e completa del gioco e apportare eventuali miglioramenti o modifiche con facilità.

Link al GitHub di Indovina la Città: https://fabiomonas.github.io/Capolavoro/

Fabio Monas